Updated Congestion Control Algorithm for TCP Throughput improvement in Wired and Wireless Network

GJCST Classifications: C.2.5, C.2.1

Prof.K.Srinivas
Dept of MCA,KIPGS,KONDAIR,509125 India; srinivas_mnb4u@yahoo.co.in
Dr.A.A.Chari
Dept of SQC,Rayalaseema University, Kurnool 518002, India; directorresearch.ru@gmail.com,
Prof.N.Kasiviswanath
Dept of CSE, GPREC, Kurnool 518002, India; nkv@gmail.co

Abstract- The basic idea proposed in this paper is to determine the Optimal Congestion Window for a TCP Sender in a particular network set-up (that corresponds to the fair share of that connection) and keep this congestion window a constant to a point where the fair share in the network has changed considerably from the instance of the calculation of the size of the last window. At this point, the TCP Congestion Window is recalculated according to the nature of new circumstances. The proposed mechanism is particularly effective over wireless links, which have an inherently loss-prone nature, as Modified TCP's congestion window being independent of packet losses (be it corruption losses or it congestion losses), keeps transmitting at the same rate at before.

I. INTRODUCTION

The well-known challenge in providing TCP congestion control algorithm [1], [2], [12] in wired – cum – wireless environment is that it relies on the packet loss as an indicator of network congestion. In order to ease the congestion scenario and to avoid a congestion collapse, a TCP Reno Sender reduces the congestion window (henceforth referred to as cwnd and expressed in number of segments) and refrains from sending packets. In the wired portion of the network, a congested router is invariably the likely reason of packet loss, while in the wireless portion a noisy, fading radio channel is the more likely cause of loss. This creates problems in TCP Reno since it does not possess the capability to distinguish and isolate congestion loss from wireless loss. Approaches to address this problem have been discussed and compared in the work by Balakrishnan et al. [3]–[4]. Three alternative approaches: end-to-end (E2E), Split Connection, and Localized Link Layer methods were carefully contrasted.

The split-connection approach [13]-[14] violates the semantics of E2E reliability. Secondly, this approach requires a lot of state maintenance at the base station.

In this paper, we propose a TCP Sender side modification of the TCP congestion control algorithm [5]. The crux of idea is that for a given network scenario, the Modified TCP Sender determines its optimal fair share of bandwidth in the link setting its cwnd in a way that it can effectively transmit with a rate that utilizes the fair share of bandwidth. After the cwnd is set to a value optimal for a given network scenario, it is kept constant to the point where the network scenario has changed by a extent significantly altering the connection's fair share. Since the value of cwnd is not decreased at any packet loss indication like retransmission on receipt of a triple DUPKT, or a coarse timeout caused by the expiration of the Retransmission Timer, hence it is not susceptible to performance degradation and cwnd reduction on the occurrences of stray packet losses. This leads to an enhanced performance in the wireless domain, as the losses are never an indication of congestion, rather they are caused due to the inherent loss-prone nature of the radio propagation medium. We provide simulation results in support to our claim that constant cwnd can outperform TCP Reno in static (i.e., certain time interval) network scenarios.

The rest of this paper is organized as follows: section 2 summarizes some related work; section 3 gives the analytical approach; section 4 describes the algorithm used by the sender; section 5 summarizes the results obtained by the simulations; section 6 gives an idea of the challenges faced while implementing such a strategy; and finally, section 7 concludes the paper.

II. RELATED WORK

NCPLD [15] compares the measured rtt with the lowest rtt (or that at the knee of the goodput – load curve). If the former is close to the latter, then the cause of a packet loss is assumed to be wireless errors. TCPW [8]–[11] measures goodput (or reception rate) and uses that rate to set the congestion window whenever a packet is detected lost. If the current goodput is below a certain band around the mean, then the cause of a packet loss is assumed congestion, otherwise the cause of loss is attributed to wireless errors.

This paper uses the TCPW bandwidth estimation scheme and compares the performance of the Modified sender with the TCPW sender. The TCPW [8]–[11] sender monitors ACKs to estimate the bandwidth currently used by, and thus available to the connection. More precisely, the sender uses (1) the ACK reception rate and (2) the information an ACK conveys regarding the amount of data delivered to the destination. The Westwood algorithm is described briefly below.

Let us assume that an ACK is received at the source at time tk, notifying that dk bytes have been received at the TCP receiver. We can measure the following sample bandwidth used by that connection as bk = dk/Δk, where Δk = tk−tk−1 and tk−1 is the time the previous ACK was received

The following discrete-time filter is used which is obtained by discretizing a continuous low-pass filter using the Tustin approximation

$$b'_k = \alpha_k b'_{k-1} + (1 - \alpha_k)(b_k + b_{k-1})/2$$

where b'_k is the filtered estimate of the available bandwidth at time $t = t_k$, $\alpha_k = (2\tau - \Delta_k)/(2\tau + \Delta_k)$, and $1/\tau$ is the cutoff frequency of the filter.

```
Algorithm after n duplicate ACKs
The pseudo code of the algorithm is the following:
if (n DUPKTs are received)
    ssthresh = (BWE * RTTmin)/seg_size;
    if (cwin > ssthresh) /* congestion avoid */
        cwin = ssthresh;
    endif
endif
```

Here, seg_size identifies the length of the payload of a TCP segment in bits.

```
Algorithm after coarse timeout expiration
The pseudo code of the algorithm is:

if (coarse timeout expires)
    ssthresh = (BWE * RTTmin)/seg_size;
    if (ssthresh < 2)
        ssthresh = 2;
    endif;
    cwin = 1;
endif
```

III. ANALYTICAL APPROACH

The logic for using a constant window would be summarized as under:

As in [1] if we measure the network load by average queue length over fixed intervals of some appropriate length, and L_i be the load at instant i, then, for a congested network we have:

$$L_i = N + \gamma L_{i-1} \qquad (1)$$

where N (a constant) accounts for the average arrival rate of the new traffic, and γLi-1 accounts for the traffic left from the last time interval. Evidently, the term γLi-1 arises when the sender is sending at a rate which is greater that its fair share leading to a fraction of packets from the previous round remaining in the network when the packets form the next round arrives in the network. But if the sender is sending at a rate that utilizes its fair share, the γLi-1 vanishes; equation (1) thereby reduces to

$$L_i = N \qquad (2)$$

which is a constant, and this forms the basis for use of a constant congestion window.

IV. TCP MODIFICATIONS

The key idea here [5] is that we can divide the entire lifetime of a TCP connection into a finite number of slots such that the connection's fair share in the network remains almost same in a particular slot, i.e. we may assume that the network scenario remains *almost static* with such slot. A change in the available share of a network, due to some connections leaving the network or some new connections joining, ends a slot and marks the beginning of the next slot. Our proposal is to use a constant TCP Congestion Window during these slots where the network scenario is assumed to remain unchanged. The beginning of a new slot would trigger a window recalculation and the cwnd would be set according to the connection's available share in that slot.

In the proposed mechanism, we use a bandwidth estimation algorithm similar to that of TCPW to obtain an estimate of the available fair share. The change in the rtt measurements is used as a trigger to move to the recalculation phase from the constant window phase (our model uses the knee region in the rtt curve as in [15] to detect a change in fare share and trigger recalculation). In our model, the Modified TCP Sender moves through three distinct phases during its lifetime: the startup phase followed by mutually interleaved window recalculation phase and constant window phase. The three phases are described with some detail as under.

A. *The Startup Phase*

At connection setup, the sender has no inkling of the network scenario. In order to impart dynamic nature, the Sender refrains from using typical default values for these essential attributes of the connection. The sender uses a slow start mechanism as in [1]. The sender continues the slow start process for say k rounds, during which it acquires various vital information about the network such as the minimum rtt measurement, a measure of the network bandwidth that the connection etc. After the first *k* rounds, the sender has acquired enough information about the network and hence calculates cwnd for the first time.

B. *Window Recalculation Phase*

When a change in available fair share is detected by the trigger, the TCP sender enters this phase. This is the most crucial phase of the connection, as in this phase, the cwnd is calculated which is kept a constant during the next phase. Hence the performance of the sender, how well it utilizes its share of the network, depends on the cwnd calculated. Along with the window recalculation process, the current value of the smoothed rtt measurements, obtained by passing the coarse rtt measurements of the individual segment through a low pass filter as suggested by Jacobson [1], is also archived for future reference.

An efficient Bandwidth Estimation Algorithm must be in place to determine the fair share of the connection in the network. The accuracy of this algorithm in determining the network share would determine the performance of the Modified TCP Sender.

C. The Constant Window Phase

During this phase of the connection, the cwnd is kept a constant irrespective of the number of ACKs received or any indications of packet loss like DUPKT or a coarse timeout. The sender keeps track of the rtt estimates from the segments that have been delivered. If the percentage change in the smoothed rtt measurements over the archives rtt measure is greater than a specified threshold, the sender exits the constant window phase and enters the Window Recalculation Phase i.e. if $|rtt_{arc}-rtt_{var}|/rtt_{arc}>\beta$, a window recalculation is made.

The algorithm's pseudo code is as follows

```
if( slow_start_state)
        slow_start(); /* open cwnd by one segment on each ACK arrival */
else
{
        if(|rtt_arc–rtt_var|/rtt_arc>β)    /*fractional increase greater than threshold */
        {
                /* recalculate window and archive the value of rtt_var */
                cwnd_ = (Estimated_Bandwidth* rtt_min) / seg_size_;
                if(cwnd_ < 1) cwnd_=1;
                        rtt_arc = rtt_var ;
        }
}
```

In the pseudo code, seg_size_ identifies the length of the TCP segments in bytes; rtt_{min} is the estimated minimum value of rtt throughout the lifetime of the particular connection, and Estimated Bandwidth is the Bandwidth Estimate obtained by some Bandwidth Estimation Algorithm.

V. PERFORMANCE ANALYSIS

In this section, we report on the basic performance behavior of the modified TCP senders and its fairness among a number of connections sharing a bottleneck link. A performance comparison is made with the TCP Reno and TCP Westwood [8]-[11] sources operating in similar network scenarios. Intermediate node buffer capacity is always set equal to the bandwidth delay product for the bottleneck link based on literature studied. Increasing the buffer capacity further does not have any impact on the performance [15]. The traffic model used is FTP with infinite data to send so that the sender has data to send whenever the network permits, and the packet size is set to 1000 bytes (1040 bytes with headers) in all experiments. The wireless subnet is error prone. In our simulations we have used the conventional TCP Sink which responds with an ACK for every packet received. There is no congestion or error in the ACK path. All simulations have been carried out for a period of 250 seconds with the TCP senders transmitting data for the entire period of simulation. All the simulations have been carried out with 802.11 MAC with a maximum available bandwidth of 1Mbps. A Two Ray Ground propagation model is used with an Omni-directional antenna. The wired subnet is error free while the wireless subnet is prone to varying error rates.

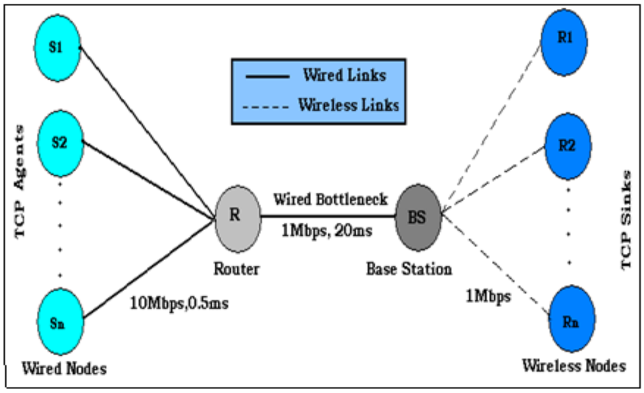

Figure 1: Network Scenario used for simulation

The performance of the Modified TCP Senders, TCP Westwood, and TCP Reno has been compared based on the throughput metric, i.e. the number of data packets received at the sender. We have analyzed the performance of the Constant Congestion Window aspect of the Modified TCP to assert that in the time slots when the share of a connection in the network remains unchanged, a Constant cwnd TCP outperforms the Reno and Westwood sources in situations with wireless errors. Our assumption is that the share of a connection remains unchanged during the entire period of simulation. The optimal cwnd for a given scenario has been evaluated the cwnd of the modified TCP Sender has been set accordingly. One aspect is to be noted that we are not simulating the entire lifetime of a modified TCP sender. Rather, our analysis is concentrated only on the Constant Congestion Window phase of the connection.

All simulations in this paper have been carried out using the LBL network simulator ns2 [6], [7] with appropriate modifications for implementation of the changes in the modified TCP sender. For comparison with TCP Westwood, the corresponding TCPW modules were used [16].

Figure 1 shows a schematic of the scenario used for simulation. A number of TCP connections share a common wired bottleneck that connects the intermediate router to the base station. When there is only one TCP connection in the network, there is no loss due to congestion. As a result, any packet loss is due to wireless errors. Hence, we can evaluate the performance of the Modified congestion control algorithm in scenarios where wireless loss is the only cause for packet loss. As the numbers of source/receiver pairs are increased, gradually the wired link between the router and the base station would become congested. Hence packets will also be lost both due to congestion as well as wireless errors. Hence, the performance of the Modified TCP sender in congested networks can also be evaluated using the same scenario.

A. *Constant Bit Error rates*

In the scenarios under consideration, the wireless subnet is prone to constant bit error rates. Figure 2 compares the performance of the Constant cwnd senders for different values of cwnd. As is evident, for every scenario, there exists a value of cwnd (in some cases more than one) for which the performance of the TCP Sender is maximum. This is the optimal cwnd for the given network scenario. In figure 2, cwnd is expressed in segments.

As is evident from figure 3, a Constant cwnd TCP outperforms the Reno and Westwood senders operating in similar network conditions. A 10-15% increase in throughput has been obtained as is evident from figure 3. In figure 3, the error rates are expressed as percentage. Figure 4 compares the performance of the TCP variants for multiple connections sharing the wired bottleneck and hence, packet is lost due to congestion as well. The Constant cwnd TCP sender outperforms Reno and Westwood in such scenarios as well.

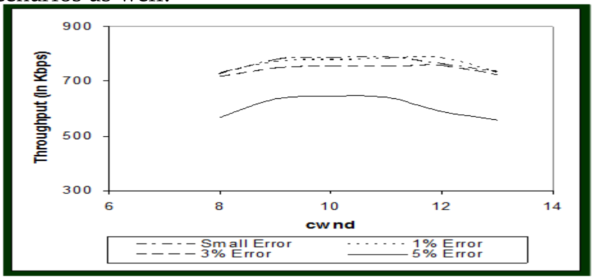

Figure 2: Variation of Throughput with varying cwnd for various bit error rates for single S/R pair.

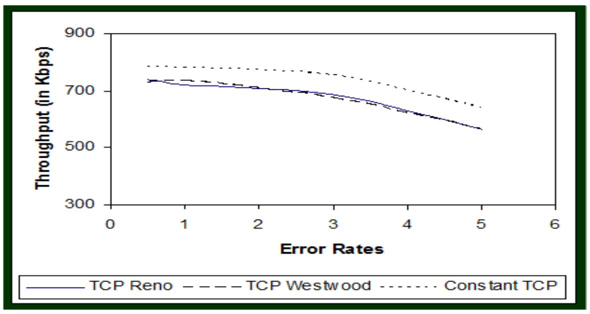

Figure 3: Variation of throughput with varying error rates in scenarios with constant bit error rates for single S/R pair

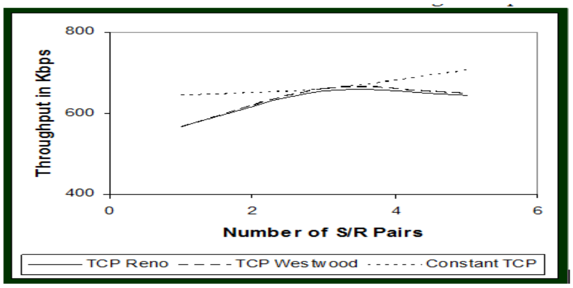

Figure 4: Variation of Throughput with number of TCP connections sharing the link in scenarios with 5% loss in constant bit error

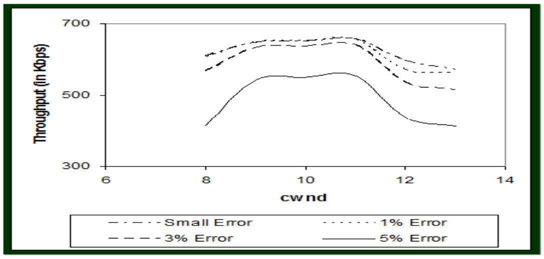

Figure 5: Variation of Throughput with varying cwnd error for various burst error for single S/R pair

B. *Burst Error*

This subsection compares the performance of Modified TCP, TCP Reno and TCP Westwood based on the throughput metric. In the scenarios under consideration, the wireless subnet is prone to burst error. The burst error is modeled using is a discrete time first order Markov Model. The pattern of errors is described by the transition matrix

$$M=\begin{bmatrix} PBB & PBG \\ PGB & PGG \end{bmatrix}$$

Where is p_{BG} the transition from bad to good, i.e., the conditional probability that successful transmission occurs in a slot given that a failure occurred in the previous slot, and the other entries in the matrix are defined similarly. It is to be noted that represents $1/(1 - p_{BB})$ the average length of a burst of errors, which is described by a geometric random variable.

Figure 5 compares the performance of the Constant cwnd senders for different values of cwnd. As is evident, for every scenario, there exists a value of cwnd (in some cases more than one) for which the performance of the TCP Sender is maximum. This is the optimal cwnd for the given network scenario. In figure 5, cwnd is expressed in segments.

When comparing the performance of the Constant cwnd TCP Sender, the cwnd is set to the optimal value for the given scenario. As is evident from figure 6, a Constant cwnd TCP outperforms the Reno and Westwood senders operating in similar network conditions. A 10-20% increase in throughput has been obtained as is evident from figure 6. In figure 6, the error rates are expressed as percentage.

Figure 7 compares the performance of the TCP variants for multiple connections sharing the wired bottleneck and hence, packet is lost due to congestion as well. The Constant cwnd TCP sender outperforms Reno and Westwood in such scenarios as well.

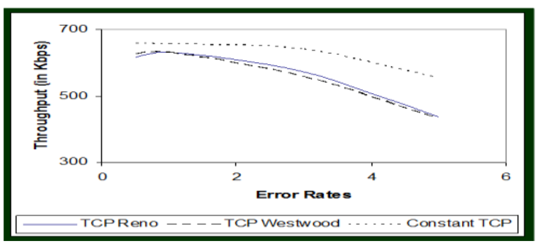

Figure 6: Variation of throughput with varying error rates in scenarios with burst error and for single S/R pair

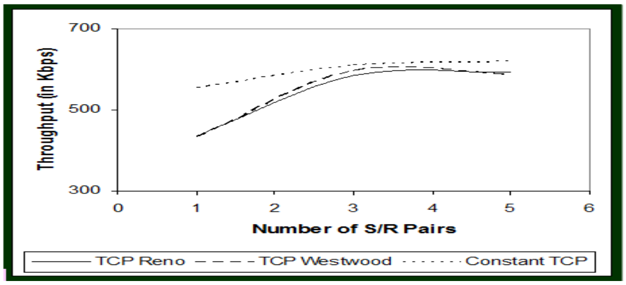

Figure 7: Variation of Throughput with number of TCP connections sharing the link in scenarios with 5% loss in burst error

VI. Challenges in Implementing the Proposed Mechanism

In the earlier sections of the paper, we have proposed Sender side modification of the TCP congestion control algorithm. There are certain challenges, which need to be met in order for this mechanism to work more efficiently. Firstly, the bandwidth determination algorithm would be precisely able to calculate the available fair share of the connection in the network. An incorrect estimation would negate the performance enhancement, which would be gained by not reducing the window in case of wireless errors. Secondly, the triggering mechanism would be able to efficiently determine a change in the available fair share of the bandwidth in the network. Failure to do so would lead to potential over or under utilization of the available fair share in case the fair share of the connection decreases or increases respectively.

VII. Conclusion and Future Work

In this paper, we propose a sender side modification of the TCP congestion control algorithm. In addition to this proposal, we have evaluated and compared the performance of the modified TCP sender during a particular phase of its lifetime viz. the Constant Congestion Window Phase. The simulations performed has shown a throughput enhancement of 10-15% as compared to TCP Reno and Westwood in cases with constant bit error rates and about 10-20% in cases with burst error corresponding to a discrete time first-order Markov model.

One important aspect of operation of this modified TCP is that the cwnd should be set to a value optimal for a given connection. For this purpose, an efficient Bandwidth Estimation Algorithm would be designed that would dynamically determine a connection’s fair share based on certain observed and measured parameters. We are working on to derive a function that would dynamically determine the cwnd during the window recalculation phase.

VIII. References

1) Jacobson and M. J. Karels, Congestion Avoidance and Control, in Proceedings of ACM SIGCOMM, August 1988, vol. 18, pp 314-329.
2) Jacobson, Modified TCP congestion avoidance algorithm. Note sent to end2end-interest mailing list, 1990.
3) H. Balakrishnan et. al., A comparison of mechanisms for improving TCP performance over wireless links, IEEE/ACM Transactions on Networking (December 1997).
4) H. Balakrishnan and R.H. Katz, Explicit loss notification and wireless web performance, in: Proceedings of IEEE GLOBECOM’98 Internet Mini-Conference, Sydney, Australia (November 1998).
5) R. Roy, S. Das, A.K. Ghosh and A. Mukherjee, "Modified TCP Congestion Control Algorithm for Throughput Enhancement in Wired-cum-Wireless Networks", Proceeding of SNCNW 2006, Luela, Sweden.
6) UCB/LBNL/VINT. The ns-2 network simulator, June 2004. http://www.isi.edu/nsnam/ ns/.
7) Fall and K. Varadhan, The ns Manual, December 13, 2003, http://www.isi.edu/nsnam/ns/ns-documentation.html
8) C Casetti, et al, TCP Westwood: End-to-End Bandwidth Estimation for Enhanced Transport over Wireless Links, Kluwer Academic Publishers. Wireless Networks 8, 467–479, 2002
9) Gerla, B.K.F. Ng, M.Y. Sanadidi, M. Valla, R. Wang, TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs, Computer Communications xx (2003) 1–18, www.elsevier.com/locate/comcom
10) J. Chen, F. Paganini, R. Wang, M.Y. Sanadidi, M. Gerla, Fluid-flow Analysis of TCP Westwood with RED, IEEE GLOBECOM 2003.
11) R. Wang, K. Yamada, M. Y. Sanadidi, and M. Gerla, TCP With Sender-Side Intelligence to Handle Dynamic, Large, Leaky Pipes, IEEE Journal on Selected Areas in Communications, Vol. 23, No. 2, Feb 2005, pp 235-248.
12) Allman, M., Paxson, V. and Stevens W.R., TCP congestion control, RFC 2581, April 1999.
13) R. Yavatkar and N. Bhagwat, Improving End – to – End Performance of TCP over Mobile Inter networks, in Proceedings of Workshop on Mobile Computing Systems and Applications, December, 1994.
14) A.Bakre and B.R. Badrinath, I-TCP: Indirect TCP for Mobile Hosts, International Conference on Distributed Computing Systems 1995, pages 136–143.
15) N. K. G. Samaraweera, Non- Congestion Packet Loss Detection for TCP error recovery using wireless links, IEE Proc. on Communications, Vol 146, No. 4, Aug 1999, pp 222 – 230.
16) Network Research Lab, Dept. of CS, UCLA, TCP Westwood modules for ns-2, http: //www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_ns2/tcp-westwood-ns2.html